Midterm Review

David Kirby

October 27, 2021

Contents

1 Performance: Metrics & Enhancement Techniques

1.1 Factors affecting execution time?

$$\text{Execution Time} = \text{IC} \times \text{CPI} \times \text{T} \tag{1}$$

- Instruction count (IC): Application/program; ISA.
- Clock per instruction (CPI): ISA; Datapath design; Parallel and pipeline HW design.
- Clock period (T): Semiconductor technology; Datapath design and implementation.
- Decrease in one may lead to increase in other two.

1.2 Challenges on performance enhancement

1.2.1 Reduction of clock cycle time (T) / increase clock frequency

- Power consumption increases with increase in clock frequency.
- Memory operations take longer than a clock period → memory-wall problem (memory is relatively slower than CPU, CPU waits for data/instructions).

1.2.2 Reduction of instruction count (IC)

- More complex instructions.
- Multi-issue processor: VLIW[1]/superscalar, SIMD[2], vector processor.

1.2.3 Reduction of cycle per instruction (CPI)

- Instruction pipelining: Pipeline datapath
- Multi-issue processor: VLIW/superscalar processor

1.3 Performance Speedup

1.3.1 Case 1: speedup of computer$_A$ over computer$_B$ can be computed as

$$\text{speedup} = \frac{\text{perf}_A}{\text{perf}_B} \tag{2}$$

$$\text{speedup} = \frac{\text{time}_B}{\text{time}_A} \tag{3}$$

[1] Very long instruction word
[2] Single instruction, multiple data

1.3.2 Case 2: speedup achieved due to enhancement can be computed as

$$\text{speedup} = \frac{\text{T}_{\text{unenhanced}}}{\text{T}_{\text{enhanced}}} \tag{4}$$

$$= \frac{\text{T}_{\text{original}}}{\text{T}_{\text{enhanced}}} \tag{5}$$

Examples: If fraction E of the program is enhanced by a factor of S in an enhanced machine, then determine the speedup of enhanced machine over the machine before enhancement (original machine). What is the maximum speedup for a given E if S can be any value greater than or equal to 1?

Fraction $U = (1 - E)$ of the program is not enhanced. If T is execution time of program in unenhanced (original) machine:

$$\text{Time required for the execution of unenhanced fraction} = \text{T} \times (1 - \text{E}) \tag{6}$$

$$\text{Time required for the execution of enhanced fraction} = \frac{(\text{T} \times \text{E})}{\text{S}} \tag{7}$$

Total execution time in the enhanced machine =

$$\begin{aligned} \text{T}' &= \left[\text{T} \times (1 - \text{E})\right] + \left[\frac{(\text{T} \times \text{E})}{\text{S}}\right] \\ &= \text{T} \times \left[(1 - \text{E}) + \frac{\text{E}}{\text{S}}\right] \\ &= \text{T} \times \left[1 - \frac{\text{E} \times (\text{S} - 1)}{\text{S}}\right] \end{aligned} \tag{8}$$

*** (check: if S = 1 then T ′ = T)

$$\begin{aligned} \text{speedup} &= \frac{\text{T}}{\text{T}'} \\ &= \frac{1}{(1 - \text{E}) + \frac{\text{E}}{\text{S}}} \end{aligned} \tag{9}$$

1.4 Amdahl's law

- The performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.
- If fraction (1−E) of application cannot be enhanced for parallel implementation, then speedup is limited by a factor of $\frac{1}{(1-\text{E})}$, even if the rest of the application is infinitely sped up, and involve infinitesimal time for the computation.
- Amdahl's law defines the speedup that can be gained by using a particular feature. What is the speedup? (see Equation (9))
- Amdahl's law gives us a quick way to find the speedup from some enhancement, which depends on two factors:

– The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement.
– The improvement gained by the enhanced execution mode, that is, how much faster the task would run if the enhanced mode were used for the entire program.

$$\text{speedup}_{\max} = \lim_{S \to \infty} \frac{1}{(1 - E)} \tag{10}$$

2 Power and Energy

2.0.1 They are NOT the same metric

- Energy is measured in Joules (J)
- Power is J/s

2.0.2 Energy per task is better metric for efficiency

- Relate to battery life in personal mobility device (PMDs)
- Reduce energy bills in WSC
- If processor A consumes $2\times$ the power as processor B but complete the same task in one-fourth the time, there is a $2\times$ gain in energy efficiency

2.1 Power: We do care

2.1.1 Power limitations

- Must get power into the IC and distribute it
- And out in the form of heat, e.g., a 2.25cm^2 die could consume 100W

2.1.2 Thermal Design Power (TDP)

- Characterizes sustained power consumption
- Used as target for power supply and cooling system
- Lower than peak power ($1.5\times$ is typical)
- Must be higher than average

2.2 Energy Consumption in CMOS

2.2.1 Dynamic vs. Static

- Dynamic: switching of transistors and clock
- Static: current leakage through imperfect transistors

2.2.2 Dynamic energy

- Consider transistor switching from 0→1 or 1→0
- A transistor gate looks like a capacitor
- Energy in a capacitor is $\frac{1}{2} \times \mathrm{C} \times \mathrm{V}^2$

2.2.3 Dynamic power

$$\mathrm{P}_{\mathrm{dynamic}} = \frac{1}{2} \times \mathrm{C} \times \mathrm{V}^2 \times \alpha \times f \tag{11}$$

*** Where α is activity factor to account switch/clock cycle

2.2.4 Static power

$$\mathrm{P}_{\mathrm{static}} = \mathrm{V} \times \mathrm{I}_{\mathrm{leakage}} \tag{12}$$

*** Where $\mathrm{I}_{\mathrm{leakage}}$ depends on the # of transistors & their leakage

2.3 Power Observations

- Dynamic power is linearly related to the clock rate and capacitance but quadratically related to voltage.
- Static power is linearly related to leakage current and voltage.
- Dennard's scaling implied that C and V of each transistor would scale down as density and f go up.
- Lower f reduces power but not energy per task! Why?
- Significant power savings can be achieved by reducing V. How do we do that?
  - Before Dennard's scaling, moving to a new tech node would get us a lower voltage.
  - Now, slowing f down allows V to be lower. Why?
  - This technique is called Dynamic Voltage Frequency Scaling (DVFS)
- Clock gating
  - Turn off the clock of the portions of unused logic
  - The clock network consumes a significant amount of power
  - Eliminates dynamic power for unused logic
  - Static power is still an issue
- Power gating
  - Turn off power of portions of unused logic
  - Eliminates both static and dynamic power
  - Dark Silicon!

- Hardware accelerate
  - Moving computationally intense portions of software into more efficient dedicated hardware
- Change state
  - Put unused cache and memory into a drowsy state where it retains contents but consumes less power by reduced voltages
- MTTF: Mean time to failure
  - MTTF is the length of time a device or other product is expected to last in operation
  - MTTF is one the many ways to evaluate the reliability of pieces of hardware or other technologies
  - If modules have independent, exponentially distributed lifetimes (age of module does not affect probability of failure), the overall failure rate is the sum of failure rates of the modules

Examples:
– MTTF is (perhaps) 100,000 hours for a fan
– MTTF is (perhaps) 1,000,000 hours for a hard disk:

$$\begin{aligned} \text{Failure rate} &= \frac{1}{\text{MTTF}} \\ &= \frac{1}{1,000,000} \end{aligned} \tag{13}$$

$$\begin{aligned} \text{MTTF} &= \frac{1}{\text{Failure rate}} \\ &= 1,000,000 \text{ [hours]} \end{aligned} \tag{14}$$

– If there are N hard disks, each with MTTF of M hours:

$$\text{Failure rate} = N \times \frac{1}{M} \tag{15}$$

– Further, a system consisting with N_1 hard disks (M_1 hours MTTF per disk), N_2 disk controller (M_2 hours MTTF per controller), and N_3 power supply (M_3 hours MTTF per supply)

$$\text{Failure rate} = N_1 \times \frac{1}{M_1} + N_2 \times \frac{1}{M_2} + N_3 \times \frac{1}{M_3} \tag{16}$$

– Assuming a system consisting of N disks (M hour MTTF per disk), and the system is considered to fail if X disks fail.

$$\text{Failure rate} = \frac{N \times \frac{1}{M}}{X} \tag{17}$$

3 Memory Hierarchy

- A key design decision is where blocks (or lines) can be placed in a cache
- Set Associative: the set is chosen by the address of the data:

  (block address) MOD (Number of sets in cache)

- Caching data that is only read is easy; caching writes is more difficult

3.1 Cache miss rate: a measure of the benefits of different cache organizations

- compulsory – initial misses due to a cold cache
- capacity – misses due to lack of associativity (i.e., too rigid of a placement)
- conflict – misses due to the small cache size
- coherency – misses due to the cache coherence protocol used for sharing memory amongst processors

3.2 Average memory access time: better measure of cache organization

$$\text{Average memory access time} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

As block size increases, so does the miss penalty, thus miss rate does not tell the whole story

3.3 Mitigation of misses

- Compulsory misses can be decreased with larger block sizes
- Conflict misses can be decreased due to associativity
- Capacity misses can be decreased with larger caches

3.4 Reducing the miss penalty

- Miss rate is only part of the AMAT equation
- As the performance gap increased, miss penalties between the L1 cache and memory became prohibitive
- Multi-level caches solved that problem

3.5 Multi-level caches

- Sandwich a larger L2 cache between the L1 and memory
- Allows L1 to be small to optimize hit time
- L2 is large enough to have a relatively low miss rate but not too large to keep access time down
- L2 access time is approximately L1 miss penalty

3.6 Miss rate for L2

3.6.1 Local miss rate

- The number of misses divided by the number of access to that particular cache
- Is often quite low for L2 cache because L1 handels most of the accesses and L2 is only accessed when L1 misses

3.6.2 Global miss rate

- The number of misses divided by the total number of memory accesses generated by the processor
- For L1, global miss rate equals the local miss rate
- For L2, global miss rate equals the local miss rate of L1 times the local miss rate of L2

3.7 Techniques that reduce Miss Rates

1. Larger Block Size to Reduce Miss Rate
    - Larger blocks take advantages of
        - The bandwidth provided by the next level in the hierarchy
        - Spatial locality
    - Larger blocks reduce compulsory misses
    - Block sized that are too large will start to suffer from conflict misses. Why?
        - Block size matters less for larger caches
        - 64 bytes seems to be the happy place
2. Larger Caches to Reduce Miss Rate
    - The obvious drawback is potentially longer hit time and higher cost and power. This technique has been especially popular in off-chip caches.
3. Higher Associativity to Reduce Miss Rate
4. Multilevel Caches to Reduce Miss Rate
    - Should I make cache faster to keep pace with speed of processors? Or make the cache larger to overcome the widening gap between the processor and main memory?
5. Giving priority to read misses over writes to reduce Miss penalty
6. Avoiding address translation during indexing of cache to reduce Hit Time.
    - Virtual Address is not built for all. Why?
        - Protection: page-level Protection
        - Cache is to be flushed
        - Duplicate addresses, called synonyms or aliases
        - I/O typically uses physical addresses

3.8 Write buffers to improve memory performance

For pipelining between L1 and L2 caches or between L2 and the memory

- Can exist anywhere in memory hierarchy
- Improves the performance by reducing the cache misses

4 Mock Exam

4.1 A designed 5-stage pipelined processor and synthesized it for a 45nm process technology node with a target clock rate of 1GHz. During power analysis, the processor is at the target clock rate with a supply voltage of 1.0V. The processor draws 70mW of dynamic power and 10mW of static power.

- The energy to complete the operations
- Scale down the clock and calculate the energy for the operations

Assuming a cryptographic operation takes 0.5 seconds to complete on your processor what is the energy per cryptographic operation at the target clock rate?

$$\begin{aligned}\text{Power}_{\text{total}} &= \text{Power}_{\text{dynamic}} + \text{Power}_{\text{static}} \\ &= 70\text{mW} + 10\text{mW} \\ &= 80\text{mW}\end{aligned}$$

$$\begin{aligned}\text{Energy per operation} &= \text{Power}_{\text{total}} \times \text{Time to complete operation} \\ &= 80\text{mW} \times \text{XX}[s] \\ &= \text{XX}[J/op]\end{aligned}$$

For certain applications your processor performs cryptographic operations 4x faster than necessary. If you were to slow the clock down to 200MHz without adjusting the voltage what would be the overall power draw? What would be the energy per cryptographic operation?

$$\begin{aligned}\text{Power}'_{\text{dynamic}} &= \text{Power}_{\text{dynamic}} \times \frac{\text{new clk rate}}{\text{original clk rate}} \\ &= 70\text{mW} \times \frac{\text{XXX MHz}}{1\text{GHz}} \\ &= \text{XX mW}\end{aligned}$$

$$\begin{aligned}\text{Power}'_{\text{total}} &= \text{Power}'_{\text{dynamic}} + \text{Power}_{\text{static}} \\ &= \text{XX mW} + 10\text{mW} \\ &= \text{XX mW}\end{aligned}$$

$$\begin{aligned}\text{Energy per operation}' &= \text{Power}'_{\text{total}} \times \text{Time to complete operation} \times \text{Speedup} \\ &= 80\text{mW} \times \text{XX}[s] \times 4\text{x faster} \\ &= \text{XX}[J/op]\end{aligned}$$

Assuming you could safely drop the voltage to 0.6V when operating at a 200MHz clock recalculate the power draw and energy per cryptographic operation. Assume the leakage current remains the same.

$$\begin{aligned}
\text{Power}'_{\text{dynamic}} &= \text{Power}_{\text{dynamic}} \times \frac{\text{new clk rate}}{\text{original clk rate}} \times \left(\frac{\text{new voltage}}{\text{original voltage}}\right)^2 \\
&= 70\text{mW} \times \frac{\text{XXX MHz}}{1\text{GHz}} \times \left(\frac{\text{XX V}}{1.0\text{V}}\right)^2 \\
&= \text{XX mW}
\end{aligned}$$

$$\begin{aligned}
\text{Power}'_{\text{static}} &= \text{Power}_{\text{static}} \times \frac{\text{new voltage}}{\text{original voltage}} \\
&= 10\text{mW} \times \frac{\text{XX V}}{1.0\text{V}} \\
&= \text{XX mW}
\end{aligned}$$

$$\begin{aligned}
\text{Power}'_{\text{total}} &= \text{Power}'_{\text{dynamic}} + \text{Power}'_{\text{static}} \\
&= \text{XX mW} + \text{XX mW} \\
&= \text{XX mW}
\end{aligned}$$

$$\begin{aligned}
\text{Energy per operation}' &= \text{Power}'_{\text{total}} \times \text{Time to complete operation} \times \text{Speedup} \\
&= \text{XX [mW]} \times \text{XX}[s] \times \text{4x faster} \\
&= \text{XX}[J/op]
\end{aligned}$$

The above questions looked at how the power and energy can be changed by varying the clock rate and voltage of a processor. Modern processors change these parameters dynamically using DVFS. What are some other techniques for reducing energy? Briefly describe these techniques.

- Clock gating turns the clock off when logic is not being used, reducing dynamic energy.
- Power gating turns the power off to unused logic saving both dynamic and static energy.
- Hardware acceleration maps computationally intense routines into more efficient hardware.

The technology node that a particular processor is fabricated with can also affect the energy efficiency. Imagine that you resynthesized your processor at a 130nm process technology node with a 1.5V supply voltage. In order to maintain the same transistor count you set your target clock rate to 200MHz. Assuming the leakage current remains the same and that the capacitance of the design approximately scales linearly with the feature size calculate the dynamic and static power for your processor at the 130nm node. What is the energy per cryptographic operation at the 130nm node and how does this compare to that of the 45nm node?

$$\begin{aligned}
\text{Power}'_{\text{dynamic}} &= \text{Power}_{\text{dynamic}} \times \frac{\text{new clk rate}}{\text{original clk rate}} \times \left(\frac{\text{new volt}}{\text{original volt}}\right)^2 \times \frac{\text{new fab}}{\text{original fab}} \\
&= 70\text{mW} \times \frac{\text{XXX MHz}}{1\text{GHz}} \times \left(\frac{\text{XX V}}{1.0\text{V}}\right)^2 \times \frac{\text{XX nm}}{45\text{ nm}} \\
&= \text{XX mW}
\end{aligned}$$

$$\begin{aligned}
\text{Power}'_{\text{static}} &= \text{Power}_{\text{static}} \times \frac{\text{new voltage}}{\text{original voltage}} \\
&= 10\text{mW} \times \frac{\text{XX V}}{1.0\text{V}} \\
&= \text{XX mW}
\end{aligned}$$

$$\begin{aligned}
\text{Power}'_{\text{total}} &= \text{Power}'_{\text{dynamic}} + \text{Power}'_{\text{static}} \\
&= \text{XX mW} + \text{XX mW} \\
&= \text{XX mW}
\end{aligned}$$

$$\begin{aligned}
\text{Energy per operation}' &= \text{Power}'_{\text{total}} \times \text{Time to complete operation} \times \text{Speedup} \\
&= \text{XX [mW]} \times \text{XX}[s] \times \text{4x faster} \\
&= \text{XX}[J/op]
\end{aligned}$$

$$\text{Comparison} = \frac{\text{Energy per operation}'}{\text{Energy per operation}}$$

4.2 MTTF – Assume a cluster has 500 computers, each of them with a MTTF of 25 days, and the failures follow an exponential distribution and are independent.

- Calculate the MTTF in different situations (refer to HW1)

$\text{MTTF} = \frac{25 \text{ days}}{\cancel{500 \text{ computers}}} \times \frac{1}{5} \times \cancel{500} = 5$ days

Under this failure model, MTTF is not dependent on the number of computers and therefore adding more computers would not have an effect on the MTTF of the cluster.

4.3 Cache misses

- Define and describe the types of cache misses we have discussed in class (three Cs). List other types of cache misses as you know.
- Global and local cache miss calculation.
- The techniques to improve cache performance. (refer to HW2 and Lecture 2)

4.4 Write buffers used in the memory hierarchy.

- How are write buffers used?
- Where are the write buffers used in the memory hierarchy?